CptS 515 Advanced Algorithms HOMEWORK-4

Sharath Kumar Karnati

November 2023

QUESTION 1:

I have k, for some k, water tanks, T1, · · · , Tk (which are identical in size and shape), whose water levels are respectively denoted by nonnegative real variables x1, · · · , xk. Without loss of generality, we assume that xi equals the amount of water that is currently in Ti. Initially, all the tanks are empty; i.e., xi = 0; $1ik$. I have m pumps p1, · · · , pm, that pump water into tanks. More precisely, a pump instruction, say, PA,c1,c2 , where $AT1, , Tk$, is to pump the same amount of water to each of the tank Ti with iA (so water levels on other tanks not in A will not change), where the amount is anywhere between c1 and c2 (including c1 and c2, of course we have assumed $0c1c2$). For instance, PT2,T5,1.5,2.4 means to pump simontaneously to T2 and T5 the same amount of water. However, the amount can be anywhere between 1.5 and 2.4. Suppose that we execute the instruction twice, say: PT2,T5,1.5,2.4;PT2,T5,1.5,2.4. The first PT2,T5,1.5,2.4 can result in 1.8 amount of water pumped into T2 and T5, respectively, and the second PT2,T5,1.5,2.4 can result in 2.15 amount of water pumped into T2 and T5, respectively. That is, the amount of water can be arbitrary chosen inside the range specified in the instruction, while the choice is independent between instructions. Now, let M be a finite state controller which is specified by a directed graph where each edge is labeled with a pump instruction. Different edges may be label with the same pump instruction and may also be labeled with different pump instructions. There is an initial node and a final node in M . Consider the following condition Bad(x1, · · · , xk): $x1 = x2 + 1 = x3 + 2x3 > x4 + 0.26$. A walk in M is a path from the initial to the final. I collect the sequence of pump instructions on the walk. If I carefully assign an amount (of wa- ter pumped) for each such pump instruction and, as a result, the water levels x1, · · · , xk at the end of the sequence of pump instruction satisfy Bad(x1, · · · , xk), then I call the walk is a bad walk. Such a walk intuitively says that there is an undesired execution of M. Design an algorithm that decides whether M has a bad walk. (Hint: first draw an example M where there is no loop and see what you can get. Then, 1 draw an M that is with a loop and see what you get. Then, draw an M that is with two nested loops and see what you get, and so on.)

SOLUTION:

From the given problem,

Given,

A Instruction P$\{T_2, T_5\}$,1.5,2.4 which means that the amount of water x which

is pumped into T_2 and T_5 will be between $1.5 < x < 2.4$.
Here we have k tanks $T_1, \ldots\ldots.., T_k$
Bad Condition :$X_1 = X_2 + 1 = X_3 + 2 \wedge X_3 > X_4 + 0.26$
M - a finite state controller (specified by a directed graph).
For this problem, we are going to use the concept of Automata-Theoretic Verification.
Procedure to find whether there is bad walk or not that satisfies the given condition:

- Here we are gonna define the Finite State Machine(FSM) nodes as a tuple which has a size k as given,$(x_1, \ldots\ldots, x_k)$.
- FSM absorbs the sequence of Pumping Commands as its language.
- Given, M be a finite state controller which will be specified by a directed graph.
- Here, we are gonna account the Bad condition as a property of system.
- So, lets consider P_bad as the language that will be used to represent the property of the system.
- Then, lets consider Pm as the language of the system which will be accepted by FSM .
- Then, here we can state that $P_M \cap \neg P_bad = \emptyset$.
- So now, we are gonna construct a transition table for the given M which is considered as a FSM.
- Here, each row will be corresponding to a transition step which will help us to find out if there is a bad walk in the M.
- Then we are gonna start traversal of graph with initial node as there is a pump instruction for every edge.
- When we are finished with the trans-versed of the whole graph and reached the end node then we are gonna check that whether the walk it is bad or not .
- Since we are given with the bad condition , then we are gonna find the rows which has a changed X_1 value.
- Then, from the shortlisted rows that has changed X_1 value, we are only gonna consider the rows which has their X_2 value also changed and satisfy the given bad condition I.e.; $X_1 = X_2 + 1$.
- We will repeat the same thing and do check rows which have X_1 value changed and X_2 condition satisfied and also satisfies the $X_1 = X_3 + 2$ condition.
- We are gonna continue shortlisting rows by the means given bad condition.
- If we have at-least one or more than one rows left after shortlisting and that if that row satisfy all the given bad condition, Then we are gonna state that the FSM will have a bad walk.

- If we find a row which have satisfied all the given bad condition then we are declare that the given FSM have a bad walk.

QUESTION 2:

The word bit comes from Shannon's work in measuring the randomness in a fair coin. However, such randomness measurement requires a probability distribution of the random variable in consideration. Suppose that a kid tosses a dice for 1000 times and hence he obtains a sequence of 1000 outcomes a1, a2, $\cdots$, a1000 where each ai is one of the six possible outcomes. Notice that a dice may not be fair at all; i.e., the probability of each outcome is not neccessarily 1/6 . Based on the sequence only, can you design an algorithm to decide how "unfair" the dice that the kid tosses is.

SOLUTION:

Given,
Sequence of random dice throws $a_1, a_2, \ldots\ldots a_1000$
Length of sequence = 1000
a_i is one of six possible outcomes.
To Do : Design an algorithm to decide how "unfair" the dice that the kid throws is.
ALGORITM:

- For this problem, we are going to use the Shannon Entropy. We are using this as a measure of the randomness for the given dice.
- Entropy provides us with the number of bits that are used to store the given information.
- So that we can measure the unfairness of the dice by doing comparison of the entropy of the given dice with the fair dice's entropy.
- Then now, from the problem the entropy of the random variable x can be given by:
  H(x) = $-\Sigma_{i=1}^{6}\ p_i\ log\ p_i$
- Here p_i is probability of seeing the ith face of the given dice when its rolled.
- We can find the p_i by doing the caluclation of the normalized frequencies that are corresponding to the ith face in the given sequence as we are given with a sequence of 1000 dice roll outcomes ($a_1, a_2, \ldots\ldots a_1000$).
- Therefore, p_i = number of ith face occurences in the sequence / 6
- In the case of fair dice, the p_i for the fair dice can be given by 1/6
- Therefore Entropy for the fair dice will be :
  H(x) = $-\Sigma_{i=1}^{6}\ p_i\ log\ p_i$
  H(x) = $-\Sigma_{i=1}^{6}\ 1/6\ log\ 1/6$
  H(x) = $log(6) \approx 0.8$

- Hence , we have the entropy of fair dice now the unfairness can be determined by measuring the difference between the entropy of unfair dice with that of the fair one we got .
- Hence, We have determined the way to determine the unfairness of the dice with the help of Shannon Entropy.

QUESTION 3:

In below, a sequence is a sequence of event symbols where each symbol is drawn from a known finite alphabet. For a sequence = a1 $\cdots$ ak that is drawn from a known finite set S of sequences, one may think it as a sequence of random variables x1 $\cdots$ xk taking values xi = ai, for each i. We assume that the lengths of the sequences in the set S are the same, say n. In mathematics, the sequence of random variables is called a stochastic process and the process may not be i.i.d at all (independent and identical distribution). Design an algorithm that takes input S and outputs the likehood on the process being i.i.d.

SOLUTION:

Given,

Sequence $\alpha = a_1, \ldots\ldots, a_k$ which is taken from a finite set (S) of sequences of variables which are given by $x_1, \ldots\ldots, x_k$ in which $x_i = a_i$, for each i .

- Length of the sequences in set S is assumed as "n".
- Here it is stated that the process may or may not be a independent and identical distribution (IID) after we had drawn a sequence from the set (S).
- For a stochastic process to be a IID , it should have mainly composed of two parts in it.
  - Firstly, to determine each of the random variable in the given sequence $x_1 \ldots\ldots\ldots, x_k$ as identical
  - Secondly, to determine each of the random variable in the given sequence $x_1, \ldots\ldots., x_k$ as independent.
- Now, we are gonna use the empirical probability distribution for each of the random variable at every time step 1,.....,k to verify whether the given distributions for all the random variable in the given sequence is identical or not identical.
- Now, lets consider the given condition, $x_i = a_i$ in which x_i accepts values from the finite set $a_1, \ldots\ldots., a_n$.
- Now we will measure the probability distribution of x_i at time t by using the normalized frequencies for each alphabet in the sequence of S at the time step t.
- Now for measuring the identical-ness, we will do the comparisons of the probabilities at each time step, or else we can do the comparison of the random variables by pairs with the help of statistical tests ,For example : Kolmogorov - Smirnov process.

- Then, now we are gonna construct a Marc hov's chain (C) with the given set (S) and we will verify that the chain C has independently distributed in order to determine each random variable's independence in the given sequence.
- If the Markov Chain C is memory less(means the probability of transitioning into future state($X_(n+1)$) with independent of states $X_1, X_2, \ldots X_n(past)$) then that marchov chain can be deemed as independent.
- Therefore, by using this process we can do the verification. (I.e., for each symbol in the alphabet we can estimate the probability distribution (P()) which can be derived from the previous step to specify that the random variables are identical).

QUESTION 4:

Let G1 and G2 be two directed graphs and v1, u1 be two nodes in G1 and v2, u2 be two nodes in G2. Suppose that from v1 to u1, there are infinitely many paths in G1 and that from v2 to u2, there are infinitely many paths in G2 as well. Design an algorithm deciding that the number of paths from v1 to u1 in G1 is "more than" the number of paths from v2 to u2 in G2, even though both numbers are infinite (but countable).

SOLUTION:

For this problem,
Given,
Two directed Graphs : G_1 and G_2
Nodes in G_1 : v_1 to u_1 and infinite paths in G_1.
Nodes in G_2 : v_2 to u_2 and infinite paths in G_1.
To Do : Design an algorithm that decides that the number of paths in G_1 is more than that of G_2 paths even though both are given infinite but countable.
ALGORITHM:
We are going to use the powers of adjacency matrices with the help of the number of paths in a graph.

- Now, firstly lets assume that for another case where we have two nodes x,y , then the number of k-length paths between these nodes will be given by $A_k[x, y]$, here lets take A as the adjacency matrix of the graph.
- Then we will use the perron's frobenius formula to find the number of paths between the given two nodes of a graph.
- Generally, it states that whether the matrices have a postive value and it also states that there always be a unique Eigen value(real) which will be greater than zero if all the matrices have positive values in a matrix and also confirms that the eigen vectors of the matrix must only have positive values.
- Then, we will create the adjacency matrix with the help of the connectivity between all the nodes in a graph.
- We will generally refer a element as A_xy when the given two nodes are connected.

- Now, we will do multiplication of A by A to get A_2, as we do it in Markov's Chain.
- We are gonna repeat this process for k times then we are gonna get A_k matrix.
- Now, here A will give us with the length and the number of paths in the graph.
- Then, we are gonna apply a infinite limit on A_k as we are given infinite paths between two nodes for both of the graphs.

$$\lim_{k \to \infty} A^k$$

- From Perron's -Frobenius , powers of an adjacency matrix which has been normalized by a spectral radius " λ " of A will approach the outer product of the particular left eigen vector " v " and right eigen vector "u".

$$\lim_{k \to \infty} \frac{1}{\lambda} * A^k = uv^T$$

- Now , we are gonna normalize the eigen vectors that are on the right hand side.
- By doing the above mentioned process ,then in the matrix that we got on the right hand side we can calculate the number of paths when we have infinite paths between two given nodes.
- So, now we are able to get the number of paths for each graph having two nodes.
- Then ,we ae gonna compare the findings of the two given graph's G_1 and G_2 by the values that we will get in the right hand side.
- From this we can state whether G_1 has more number of paths than compared to that of the G_2.